Unit: 2 Basic Attributes and Events of Important Android Widgets (UI)

2.1.1 Android ListView

Android **ListView** is a view which contains the group of items and displays in a scrollable list. ListView is implemented by importing *android.widget.ListView* class. ListView is a default scrollable which does not use other scroll view.

ListView uses Adapter classes which add the content from data source (such as string array, array, database etc) to ListView. Adapter bridges data between an *AdapterViews* and other Views (ListView, ScrollView etc).

Example of ListView

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>

   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      tools:context=".ListViewExample"
      android:orientation="vertical">

      <ListView
        android:id="@+id/listview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

  </LinearLayout>
```

Activity class

In java class we need to add adapter to listview using setAdapter() method of listview.

File: MainActivity.java

```
package com.example.widgets;

   import androidx.appcompat.app.AppCompatActivity;
   import android.os.Bundle;


`
```

```
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;

public class ListViewExample extends AppCompatActivity {

    ListView listView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_view_example);
        listView=(ListView)findViewById(R.id.listview);

        final ArrayList<String> arrayList=new ArrayList<>();

        arrayList.add("India");
        arrayList.add("America");
        arrayList.add("China");
        arrayList.add("Japan");
        arrayList.add("Nepal");
       arrayList.add("Pakistan");
        arrayList.add("Russia");
       arrayList.add("Australia");
        arrayList.add("Dubai");
        arrayList.add("Saudi Arabia");
        arrayList.add("Qatar");
        arrayList.add("Afghanistan");
        arrayList.add("Bangladesh");
        arrayList.add("Srilanka");
        arrayList.add("Africa");
        arrayList.add("Germany");
        arrayList.add("NewZealand");
        arrayList.add("Portugle");
```

`

```
        arrayList.add("Bahrain");
        arrayList.add("Belgium");

        ArrayAdapter<String> arrayAdapter=new ArrayAdapter<String>(getApplicationContext(),
R.layout.support_simple_spinner_dropdown_item,arrayList);

        listView.setAdapter(arrayAdapter);

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
          @Override
          public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            Toast.makeText(
                        getApplicationContext(),
                        "country: "+arrayList.get(position),
                        Toast.LENGTH_SHORT).show();
          }
        });
    }
  }
```

2.1.2 Android Custom ListView (Adding Images, sub-title)

After creating simple ListView, android also provides facilities to customize our ListView.

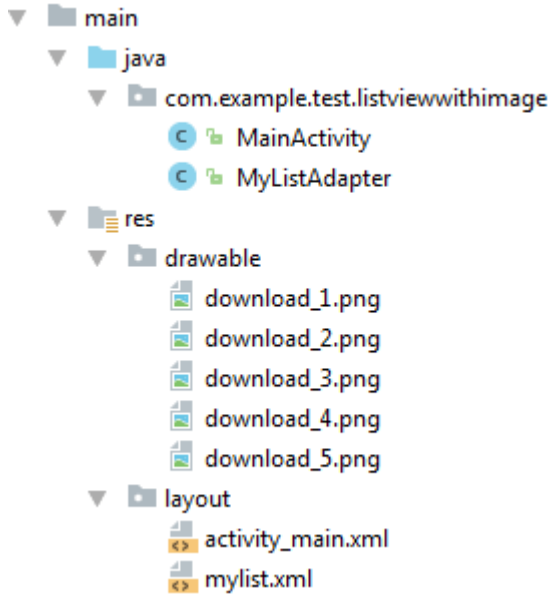
As the simple ListView, custom ListView also uses Adapter classes which added the content from data source (such as string array, array, database etc). Adapter bridges data between an AdapterViews and other Views

Example of Custom ListView

In this custom listview example, we are adding image, text with title and its sub-title.

`

Structure of custom listview project



activity_main.xml

Create an activity_main.xml file in layout folder.

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context="com.example.test.listviewwithimage.MainActivity">


  <ListView
    android:id="@+id/list"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="50dp">
  </ListView>
```

`

```
</RelativeLayout>
```

Create an additional mylist.xml file in layout folder which contains view components displayed in listview.

mylist.xml

File: mylist.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="horizontal" >

  <ImageView
    android:id="@+id/icon"
    android:layout_width="60dp"
    android:layout_height="60dp"
    android:padding="5dp" />

  <LinearLayout android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">

  <TextView
    android:id="@+id/title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Medium Text"
    android:textStyle="bold"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="5dp"
    android:padding="2dp"
    android:textColor="#4d4d4d" />
  <TextView
    android:id="@+id/subtitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

`

```
        android:text="TextView"
        android:layout_marginLeft="10dp"/>
    </LinearLayout>
</LinearLayout>
```

Place the all required images in drawable folder.

Activity class

File: MainActivity.java

```
package com.example.test.listviewwithimage;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    ListView list;

    String[] maintitle ={
            "Title 1","Title 2",
            "Title 3","Title 4",
            "Title 5",
    };

    String[] subtitle ={
            "Sub Title 1","Sub Title 2",
            "Sub Title 3","Sub Title 4",
            "Sub Title 5",
    };

    Integer[] imgid={
            R.drawable.download_1,R.drawable.download_2,
            R.drawable.download_3,R.drawable.download_4,
            R.drawable.download_5,
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

`

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        MyListAdapter adapter=new MyListAdapter(this, maintitle, subtitle,imgid);
        list=(ListView)findViewById(R.id.list);
        list.setAdapter(adapter);


        list.setOnItemClickListener(new AdapterView.OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View view,int position, long id) {
                // TODO Auto-generated method stub
                if(position == 0) {
                    //code specific to first list item
                    Toast.makeText(getApplicationContext(),"Place Your First Option Code",Toast.LE
NGTH_SHORT).show();
                }

                else if(position == 1) {
                    //code specific to 2nd list item
                    Toast.makeText(getApplicationContext(),"Place Your Second Option Code",Toast
.LENGTH_SHORT).show();
                }

                else if(position == 2) {

                    Toast.makeText(getApplicationContext(),"Place Your Third Option Code",Toast.L
ENGTH_SHORT).show();
                }
                else if(position == 3) {

                    Toast.makeText(getApplicationContext(),"Place Your Forth Option Code",Toast.L
ENGTH_SHORT).show();
                }
                else if(position == 4) {

                    Toast.makeText(getApplicationContext(),"Place Your Fifth Option Code",Toast.LE
NGTH_SHORT).show();
                }

            }
        });
    }
```

`

```
}
```

Customize Our ListView

Create another java class MyListView.java which extends ArrayAdapter class. This class customizes our listview.

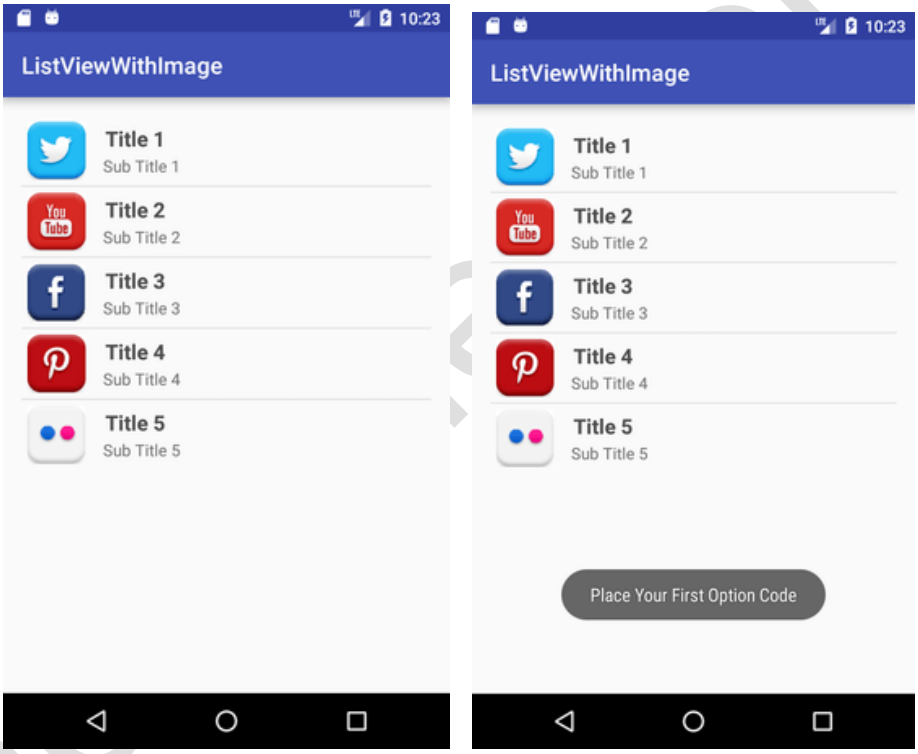
MyListView.java

```
package com.example.test.listviewwithimage;

import android.app.Activity;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class MyListAdapter extends ArrayAdapter<String> {

   private final Activity context;
   private final String[] maintitle;
   private final String[] subtitle;
   private final Integer[] imgid;

   public MyListAdapter(Activity context, String[] maintitle,String[] subtitle, Integer[] imgid) {

     super(context, R.layout.mylist, maintitle);
     // TODO Auto-generated constructor stub

     this.context=context;
     this.maintitle=maintitle;
     this.subtitle=subtitle;
     this.imgid=imgid;

   }

   public View getView(int position,View view,ViewGroup parent) {
```

`

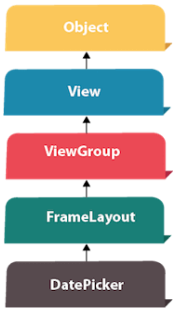
```
        LayoutInflater inflater=context.getLayoutInflater();
        View rowView=inflater.inflate(R.layout.mylist, null,true);

        TextView titleText = (TextView) rowView.findViewById(R.id.title);
        ImageView imageView = (ImageView) rowView.findViewById(R.id.icon);
        TextView subtitleText = (TextView) rowView.findViewById(R.id.subtitle);

        titleText.setText(maintitle[position]);
        imageView.setImageResource(imgid[position]);
        subtitleText.setText(subtitle[position]);

        return rowView;

    };
}
```

Output



`

2.2.1 Android DatePicker Example



Android DatePicker is a widget to select date. It allows you to select date by day, month and year. Like DatePicker, android also provides TimePicker to select time.

The android.widget.DatePicker is the subclass of FrameLayout class.

Android DatePicker Example

Let's see the simple example of datepicker widget in android.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      tools:context=".DatePickerExample"
      android:padding="16dp"
      android:orientation="vertical"
      android:gravity="center">

     <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <DatePicker
           android:id="@+id/datepicker1"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:layout_gravity="center"
```

`

```
        />

    </ScrollView>
</LinearLayout>
```

Activity class

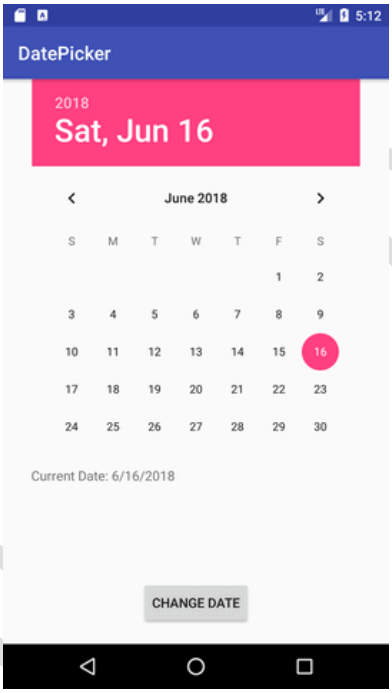
File: MainActivity.java

```
package com.example.widgets;

  import androidx.appcompat.app.AppCompatActivity;
  import android.os.Bundle;
  import android.widget.DatePicker;
  import android.widget.Toast;

  import java.util.Calendar;

 public class DatePickerExample extends AppCompatActivity {
     DatePicker datePicker;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_date_picker_example);

      datePicker=(DatePicker)findViewById(R.id.datepicker1);

      Calendar calendar=Calendar.getInstance();
          datePicker.init(calendar.get(Calendar.YEAR),
                          calendar.get(Calendar.MONTH),
                          calendar.get(Calendar.DAY_OF_MONTH),
                          new DatePicker.OnDateChangedListener() {
        @Override
        public void onDateChanged(DatePicker view, int year, int monthOfYear, int
dayOfMonth) {
          monthOfYear=monthOfYear+1;
          Toast.makeText(getApplicationContext(),

`
```

```
                "Date selected: "+dayOfMonth+"/"+monthOfYear+"/"+year,
                Toast.LENGTH_SHORT).show();
        }
    });

  }
}
```

XML attributes

android:divider	Drawable or color to draw between list items.
android:dividerHeight	Height of the divider.
android:entries	Reference to an array resource that will populate the ListView.
android:footerDividersEnabled	When set to false, the ListView will not draw the divider before each footer view.
android:headerDividersEnabled	When set to false, the ListView will not draw the divider after each header view.



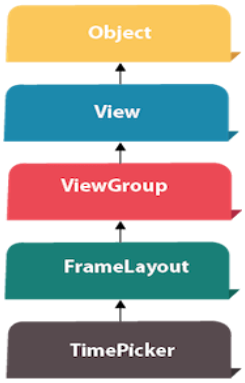
`

XML attributes

XML attributes	
android:calendarTextColor	The text color list of the calendar.
android:calendarViewShown	Whether the calendar view is shown.
android:datePickerMode	Defines the look of the widget.
android:dayOfWeekBackground	The background color for the header's day of week.
android:dayOfWeekTextAppearance	The text color for the header's day of week.
android:endYear	The last year (inclusive), for example "2010".
android:firstDayOfWeek	The first day of week according to Calendar._
android:headerBackground	The background for the selected date header.
android:headerDayOfMonthTextAppearance	The text appearance for the day of month (ex.
android:headerMonthTextAppearance	The text appearance for the month (ex.
android:headerYearTextAppearance	The text appearance for the year (ex.
android:maxDate	The maximal date shown by this calendar view in mm/dd/yyyy format.
android:minDate	The minimal date shown by this calendar view in mm/dd/yyyy format.
android:spinnersShown	Whether the spinners are shown.
android:startYear	The first year (inclusive), for example "1940".
android:yearListItemTextAppearance	The list year's text appearance in the list.
android:yearListSelectorColor	The list year's selected circle color in the list.

`

2.2.2 Android TimePicker Example



Android TimePicker widget is used to select date. It allows you to select time by hour and minute. You cannot select time by seconds.

The android.widget.TimePicker is the subclass of FrameLayout class.

Android TimePicker Example

Let's see a simple example of android time picker.

activity_main.xml
File: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".TimePickerExample"
    android:padding="16dp"
    android:orientation="vertical"
    android:gravity="center">

   <ScrollView
     android:layout_width="match_parent"
     android:layout_height="wrap_content">

     <TimePicker
        android:id="@+id/timepicker1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
```

`

```
    </ScrollView>

  </LinearLayout>
```

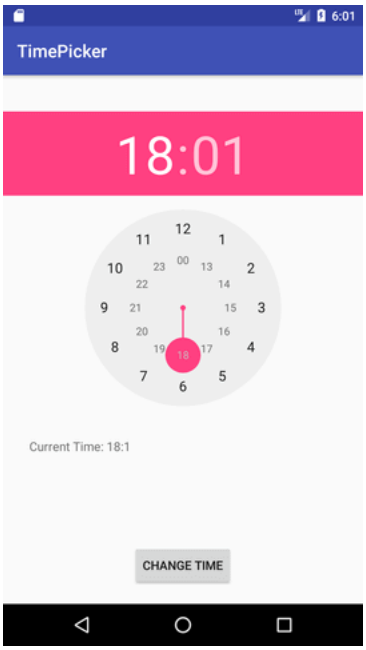
Activity class
File: MainActivity.java

```
package com.example.widgets;

   import androidx.appcompat.app.AppCompatActivity;
   import android.os.Bundle;
   import android.widget.TimePicker;
   import android.widget.Toast;

   public class TimePickerExample extends AppCompatActivity {

     TimePicker timePicker;

     @Override
     protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_time_picker_example);

       timePicker=(TimePicker)findViewById(R.id.timepicker1);
       timePicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener() {
         @Override
         public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
          Toast.makeText(getApplicationContext(),
                                "Time selected: "+hourOfDay+":"+minute,
                                Toast.LENGTH_SHORT).show();
         }
       });

     }
```

`

}

Output



XML attributes

android:timePickerMode

Defines the look of the widget. Prior to the L release, the only choice was spinner. As of L, with the Material theme selected, the default layout is clock, but this attribute can be used to force spinner to be used instead.

Must be one of the following constant values.

Constant	Value	Description
clock	2	Time picker with clock face to select the time.
spinner	1	Time picker with spinner controls to select the time.

`

2.2.3 Android ProgressBar Example

We can display the **android progress bar** dialog box to display the status of work being done e.g. downloading file, analyzing status of work etc.

In this example, we are displaying the progress dialog for dummy file download operation.

Here we are using **android.app.ProgressDialog** class to show the progress bar. Android ProgressDialog is the subclass of AlertDialog class.

The **ProgressDialog** class provides methods to work on progress bar like setProgress(), setMessage(), setProgressStyle(), setMax(), show() etc. The progress range of Progress Dialog is 0 to 10000.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      tools:context=".ProgressBarExample"
     android:gravity="center"
     android:orientation="vertical"
     android:padding="16dp">

     <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click on start button"
        android:layout_marginBottom="20dp"/>
```

`

```
    <ProgressBar
        android:id="@+id/progress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp"
        style="@style/Widget.AppCompat.ProgressBar.Horizontal"
        android:max="100"
        />

    <Button
        android:id="@+id/start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start"
        android:textColor="@color/colorWhite"
        android:background="@color/colorOrange"
        />

  </LinearLayout>
```

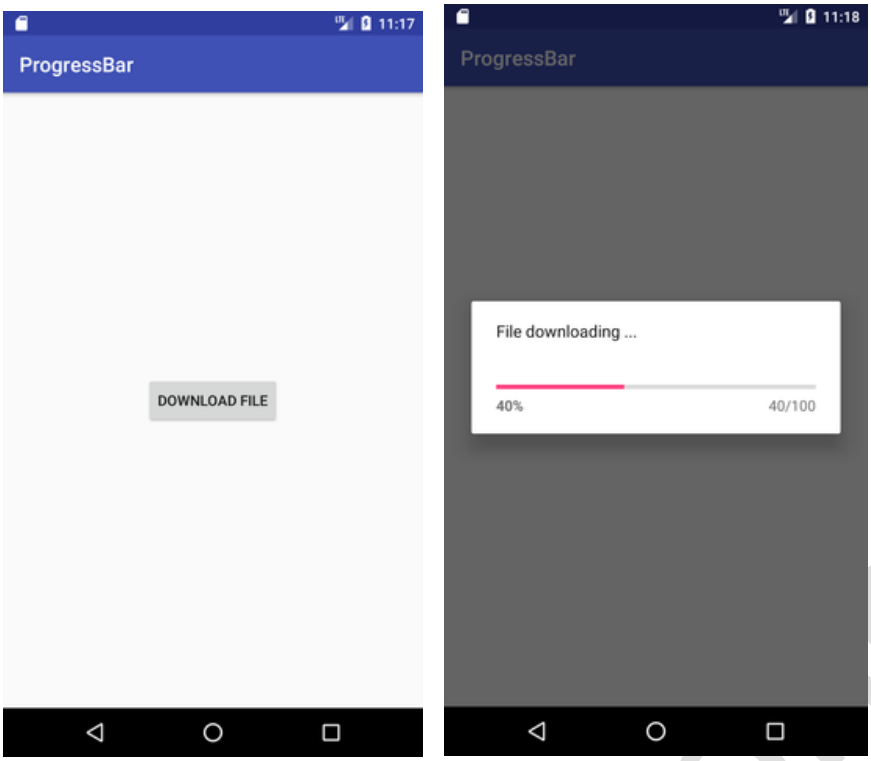
Activity class

```
  package com.example.widgets;

  import android.os.Handler;
  import androidx.appcompat.app.AppCompatActivity;
  import android.os.Bundle;
  import android.view.View;
  import android.widget.Button;
  import android.widget.ProgressBar;
  import android.widget.TextView;

  public class ProgressBarExample extends AppCompatActivity {

    TextView textView;
    ProgressBar progressBar;
```

`

```
Button button;
Handler handler;
int progresscount=0;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_progress_bar_example);

   textView=(TextView)findViewById(R.id.text);
   progressBar=(ProgressBar)findViewById(R.id.progress);
   button=(Button)findViewById(R.id.start);

   button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {

         handler=new Handler();
         handler.postDelayed(new Runnable() {
            @Override
            public void run() {
               if (progressBar.getProgress()<100){
                  progressBar.setProgress(progresscount);
                  progresscount++;
                  handler.postDelayed(this,100);
                  textView.setText("Plz wait...");
               }else{
                  textView.setText("completed...");
               }
            }
         },100);
      }
   });

   }
}
```

`



android:animationResolution	Timeout between frames of animation in milliseconds.
android:indeterminate	Allows to enable the indeterminate mode.
android:indeterminateBehavior	Defines how the indeterminate mode should behave when the progress reaches max.
android:indeterminateDrawable	Drawable used for the indeterminate mode.
android:indeterminateDuration	Duration of the indeterminate animation.
android:indeterminateOnly	Restricts to ONLY indeterminate mode (state-keeping progress mode will not work).
android:indeterminateTint	Tint to apply to the indeterminate progress indicator.
android:indeterminateTintMode	Blending mode used to apply the indeterminate progress indicator tint.
android:interpolator	Sets the acceleration curve for the indeterminate animation.
android:max	Defines the maximum value.
android:maxHeight	An optional argument to supply a maximum height for this view.
android:maxWidth	An optional argument to supply a maximum width for this view.
android:min	Defines the minimum value.

`

android:minHeight	
android:minWidth	
android:mirrorForRtl	Defines if the associated drawables need to be mirrored when in RTL mode.
android:progress	Defines the default progress value, between 0 and max.
android:progressBackgroundTint	Tint to apply to the progress indicator background.
android:progressBackgroundTintMode	Blending mode used to apply the progress indicator background tint.
android:progressDrawable	Drawable used for the progress mode.
android:progressTint	Tint to apply to the progress indicator.
android:progressTintMode	Blending mode used to apply the progress indicator tint.

`

2.3 Horizantal and Vertical ScrollView

Horizantal ScrollView

- A **HorizontalScrollView** is a *FrameLayout*.
- The **android.widget.HorizontalScrollView** class provides the functionality of horizontal scroll view.
- HorizontalScrollView is used to scroll the child elements or views in a horizontal direction. HorizontalScrollView only supports horizontal scrolling.

Horizantal ScrollView Example

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:text="Horizontal ScrollView Example"
    android:id="@+id/textView"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true" />

  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="25dp">

    <HorizontalScrollView
      android:layout_width="match_parent"
      android:layout_height="60dp"
      android:id="@+id/horizontalScrollView">

      <LinearLayout
        android:layout_width="wrap_content"
```

`

```
android:layout_height="wrap_content"
android:orientation="horizontal">

<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="New Button1"
   android:id="@+id/button1" />

<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="New Button2"
   android:id="@+id/button2" />

<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="New Button3"
   android:id="@+id/button3" />

<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="New Button4"
   android:id="@+id/button4" />

<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="New Button5"
   android:id="@+id/button5" />

<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="New Button6"
   android:id="@+id/button6" />
```

`

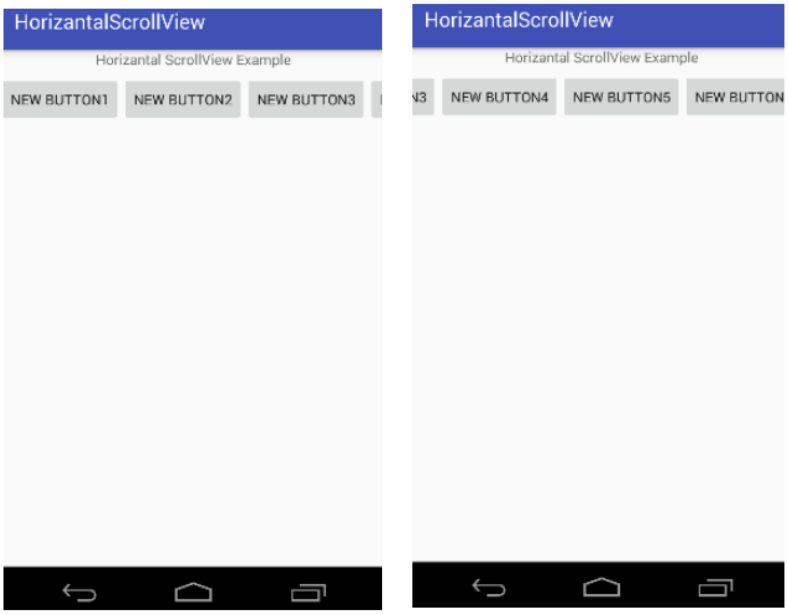
```
            <Button
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="New Button7"
               android:id="@+id/button7" />

            <Button
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="New Button8"
               android:id="@+id/button8"/>
         </LinearLayout>

      </HorizontalScrollView>
   </LinearLayout>
</RelativeLayout>
```

MainActivity.java

```
package com.example.test.horizantalscrollview;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
}
```

OUTPUT

`



Vertical ScrollView

- The **android.widget.ScrollView** class provides the functionality of scroll view. ScrollView is used to scroll the child elements of palette inside ScrollView.
- Android supports vertical scroll view as default scroll view. Vertical ScrollView scrolls elements vertically.

Vertical ScrollView Example

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.example.test.scrollviews.MainActivity">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

`

```
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Vertical ScrollView example"
    android:id="@+id/textView"
    android:layout_gravity="center_horizontal"
    android:layout_centerHorizontal="true"
    android:layout_alignParentTop="true" />


<ScrollView android:layout_marginTop="30dp"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/scrollView">

   <LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="vertical" >

      <Button
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Button 1" />

  <Button
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Button 2" />

  <Button
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Button 3" />


   <Button
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
```

`

```
    android:text="Button 4" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 5" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 6" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 7" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 8" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 9" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 10" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 11" />
<Button
    android:layout_width="fill_parent"
```

`

```
    android:layout_height="wrap_content"
    android:text="Button 12" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 13" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 14" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 15" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 16" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 17" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 18" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 19" />
```

`

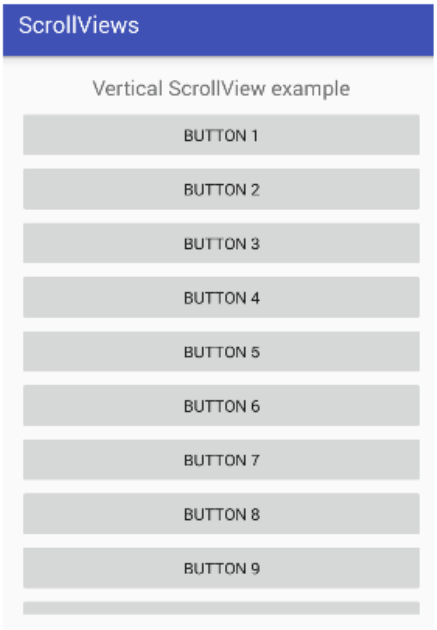
```
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 20" />

      </LinearLayout>
    </ScrollView>
  </RelativeLayout>
```

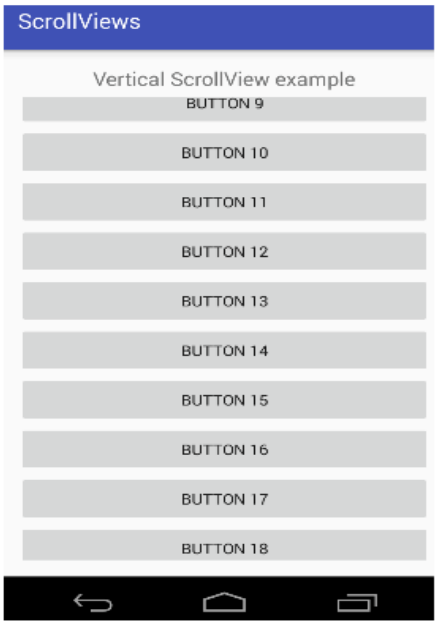
MainActivity.java

```
package com.example.test.scrollviews;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
}
```

OUTPUT

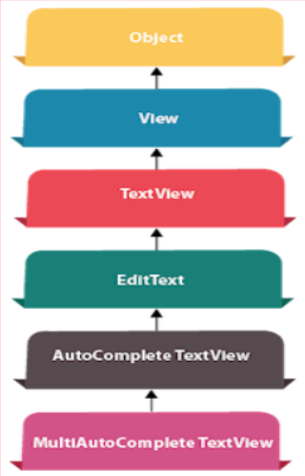
`





`

2.4.1Android AutoCompleteTextView Example



Android AutoCompleteTextView

completes the word based on the reserved words, so no need to write all the characters of the word. Android AutoCompleteTextView is a editable text field, it displays a list of suggestions in a drop down menu from which user can select only one suggestion or value.

Android AutoCompleteTextView is the subclass of EditText class. The MultiAutoCompleteTextView is the subclass of AutoCompleteTextView class.

Android AutoCompleteTextView Example

In this example, we are displaying the programming languages in the autocompletetextview. All the programming languages are stored in string array. We are using the **ArrayAdapter** class to display the array content.

Let's see the simple example of autocompletetextview in android.

activity_main.xml

Drag the AutoCompleteTextView and TextView from the pallete, now the activity_main.xml file will like this:

File: activity_main.xml

<?xml version="1.0" encoding="utf-8"**?>**

`

```
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.javatpoint.com.autocompletetextview.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="What is your favourite programming language?"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.032" />

    <AutoCompleteTextView
        android:id="@+id/autoCompleteTextView"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="92dp"
        android:layout_marginTop="144dp"
        android:text=""
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Activity class

Let's write the code of AutoCompleteTextView.

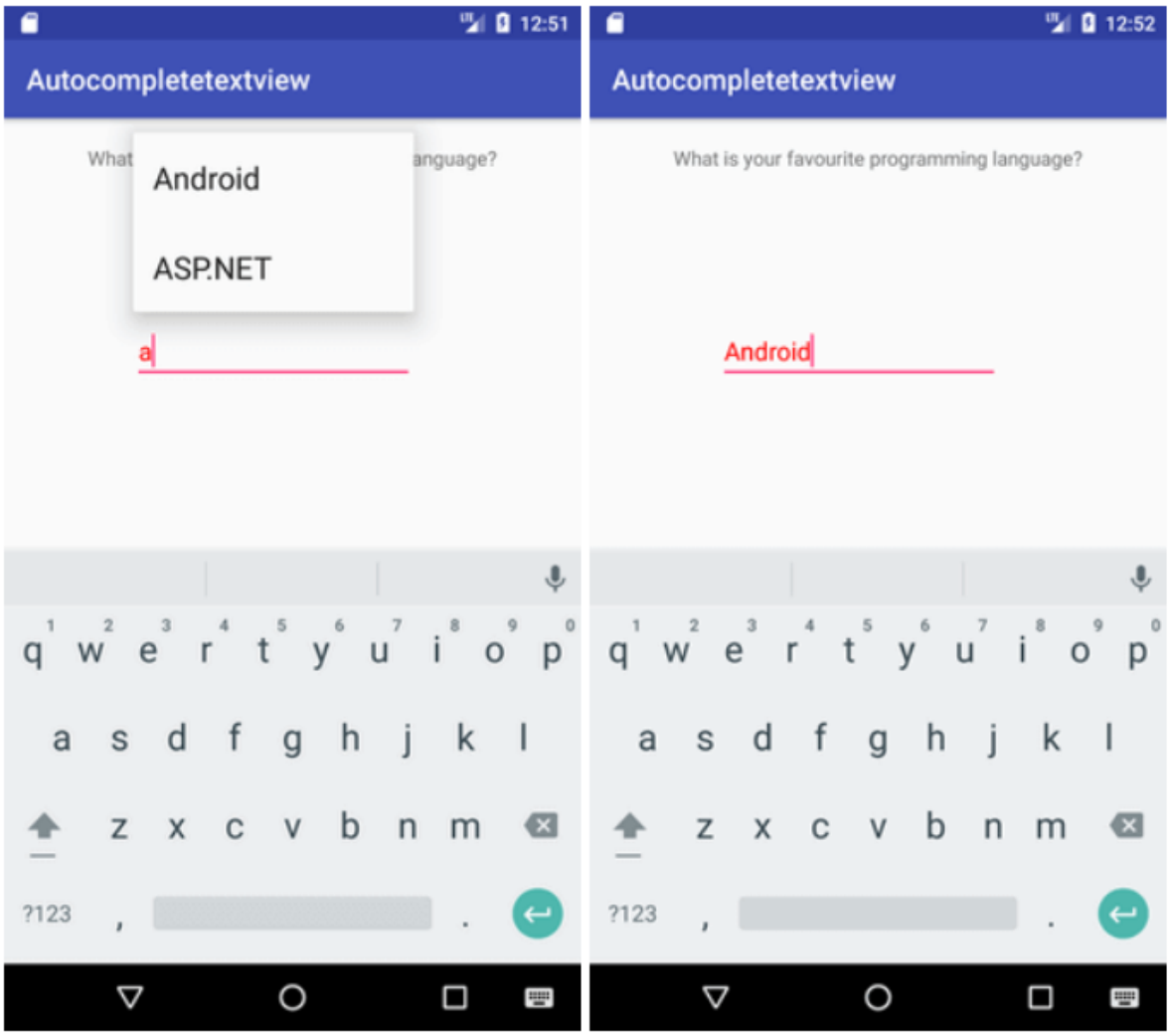
File: MainActivity.java

```
`
```

```
package example.javatpoint.com.autocompletetextview;

import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

public class MainActivity extends AppCompatActivity {
   String[] language ={"C","C++","Java",".NET","iPhone","Android","ASP.NET","PHP"};
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      //Creating the instance of ArrayAdapter containing list of language names
      ArrayAdapter<String> adapter = new ArrayAdapter<String>
            (this,android.R.layout.select_dialog_item,language);
      //Getting the instance of AutoCompleteTextView
      AutoCompleteTextView actv =  (AutoCompleteTextView)findViewById(R.id.autoComplet
eTextView);
      actv.setThreshold(1);//will start working from first character
      actv.setAdapter(adapter);//setting the adapter data into the AutoCompleteTextView
      actv.setTextColor(Color.RED);
   }
}
```

`

Output:



2.4.2 Android EditText with TextWatcher (Searching data from ListView)

Android **EditText** is a subclass of *TextView*. EditText is used for entering and modifying text. While using EditText width, we must specify its input type in *inputType* property of EditText which configures the keyboard according to input.

EditText uses **TextWatcher** interface to watch change made over EditText. For doing this, EditText calls the *addTextChangedListener()* method.

Methods of TextWatcher

`

1. **beforeTextChanged(CharSequence s, int start, int count, int after):** It is executed before making any change over EditText. Within **s**, the **count** characters beginning at **start** are about to be replaced by new text with length **after**.
2. **onTextChanged(CharSequence cs, int start, int before, int count):** It is executed while making any change over EditText. Within **cs**, the **count** characters beginning at **start** have just replaced old text that had length **before**.
3. **afterTextChanged(Editable s):** It is executed after change made over EditText.

Example of EditText with TextWatcher()

In this example, we will implement EditText with TextWatcher to search data from ListView.

activity_main.xml

Create an activity_main.xml file in layout folder containing EditText and ListView.

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.example.test.searchfromlistview.MainActivity">

   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/editText"
      android:inputType="text"
      android:layout_alignParentTop="true"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true" />
```

`

```
    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_below="@+id/editText"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

Create another file list_item.xml in layout folder which contains data of ListView.

list_item.xml

File: list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

<TextView android:id="@+id/product_name"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="10dip"
    android:textSize="16dip"
    android:textStyle="bold"/>
</LinearLayout>
```

`

Activity class

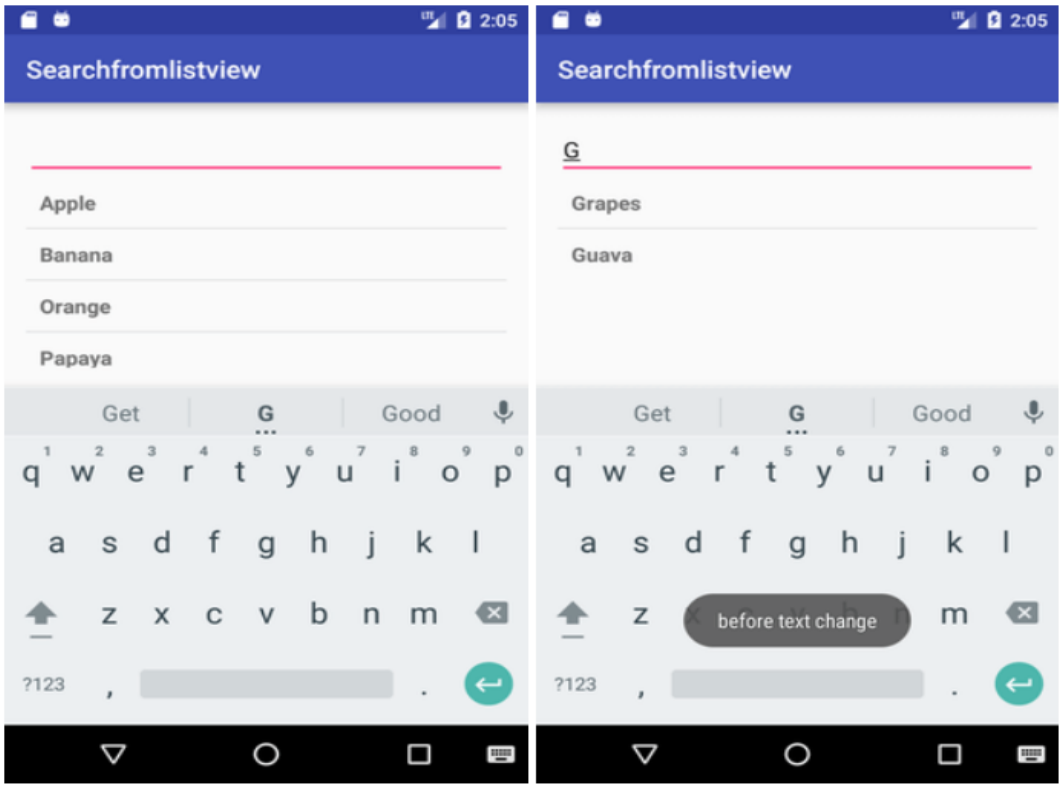
Activity class

```
package com.example.test.searchfromlistview;

import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

   private ListView lv;
   private EditText editText;
   private ArrayAdapter<String> adapter;

   private String products[] = {"Apple", "Banana","Pinapple", "Orange", "Papaya", "Melon", "Gra
pes", "Water Melon","Lychee", "Guava", "Mango", "Kivi"};
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      lv = (ListView) findViewById(R.id.listView);
      editText = (EditText) findViewById(R.id.editText);
      adapter = new ArrayAdapter<String>(this, R.layout.list_item, R.id.product_name, products)
;
      lv.setAdapter(adapter);

      editText.addTextChangedListener(new TextWatcher() {

`
```

```
        @Override
        public void onTextChanged(CharSequence cs, int arg1, int arg2, int arg3) {
           adapter.getFilter().filter(cs);
        }

        @Override
        public void beforeTextChanged(CharSequence arg0, int arg1, int arg2, int arg3) {
           Toast.makeText(getApplicationContext(),"before text change",Toast.LENGTH_LONG).s
how();
        }

        @Override
        public void afterTextChanged(Editable arg0) {
           Toast.makeText(getApplicationContext(),"after text change",Toast.LENGTH_LONG).sho
w();
        }
      });
   }
}
```

Output:

`



2.5 Image Slider, ImageSwitcher, Search View

Image Slider

- Android *image slider* slides one entire screen to another screen. Image slider is created by **ViewPager** which is provided by support library. To implement image slider, you need to inherit ViewPager class which extends PagerAdapter.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <androidx.viewpager.widget.ViewPager
      android:layout_width="match_parent"
      android:layout_height="match_parent"
```

`

```
    android:id="@+id/viewpager"/>
</RelativeLayout>
```

MainActivity.java

```
package com.example.imageslider;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;
import androidx.viewpager.widget.ViewPager;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ViewPager mViewPager = findViewById(R.id.viewpager);
    ImageAdapter adapterView = new ImageAdapter(this);
    mViewPager.setAdapter(adapterView);
  }
}
```

OUTPUT

`

Methods of PagerAdapter Class

isViewFromObject(View, Object)	This method checks the view whether it is associated with key and returned by instantiateItem().
instantiateItem(ViewGroup, int)	This method creates the page position passed as an argument.
destroyItem(ViewGroup, int, Object)	It removes the page from its current position from container. In this example we simply removed object using removeView().
getCount()	It returns the number of available views in ViewPager.

Image Switcher

- Sometimes you don't want an image to appear abruptly on the screen, rather you want to apply some kind of animation to the image when it transitions from one image to another. This is supported by android in the form of ImageSwitcher.
- An image switcher allows you to add some transitions on the images through the way they appear on screen. In order to use image Switcher, you need to define its XML component first.

activity_main.xml

`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context=".MainActivity">

   <ImageSwitcher
      android:layout_width="match_parent"
      android:layout_height="300dp"
      android:layout_gravity="center"
      android:id="@+id/imageswitcher"/>

<Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:id="@+id/btnswitch"
      android:text="Switch"/>

</LinearLayout>
```

MainActivity.java

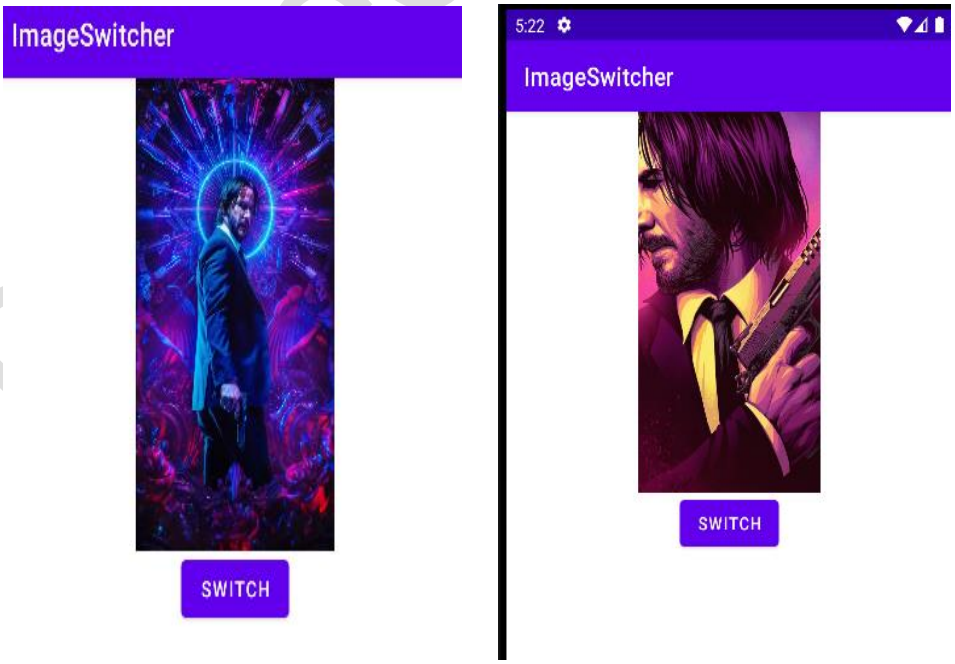
```
package com.example.imageswitcher;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.ViewSwitcher;



`
```

```
public class MainActivity extends AppCompatActivity {

   ImageSwitcher imageSwitcher;
   Button button;
   int imageIdList[]
={R.drawable.image1,R.drawable.image2,R.drawable.image3,R.drawable.image4,R.drawable.i
mage5};
   int count = imageIdList.length;
   int currentIndex = -1;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      imageSwitcher = findViewById(R.id.imageswitcher);
      button = findViewById(R.id.btnswitch);
      imageSwitcher.setFactory(new ViewSwitcher.ViewFactory() {
         @Override
         public View makeView() {
            ImageView myView = new ImageView(getApplicationContext());
            myView.setScaleType(ImageView.ScaleType.FIT_CENTER);
            myView.setLayoutParams(new
            ImageSwitcher.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
            ViewGroup.LayoutParams.WRAP_CONTENT));
            return myView;
         }
      });
      button.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            currentIndex++;
            if (currentIndex==count)
                  currentIndex = 0;
            imageSwitcher.setImageResource(imageIdList[currentIndex]);
         }
      });

   }
}


`
```

Java Methods of ImageSwicher

setImageDrawable(Drawable drawable)	Sets an image with image switcher. The image is passed in the form of bitmap
setImageResource(int resid)	Sets an image with image switcher. The image is passed in the form of integer id
setImageURI(Uri uri)	Sets an image with image switcher. THe image is passed in the form of URI

OUTPUT



`

Search View

- Android **SearchView** provides user interface to search query submitted over search provider.
- SearchView widget can be implemented over ToolBar/ActionBar or inside a layout.
- SearchView is by default collapsible and set to be iconified using *setIconifiedByDefault(true)* method of SearchView class. For making search field visible, SearchView uses setIconifiedByDefault(false) method.

Search View Example

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context=".MainActivity">

   <SearchView
      android:layout_marginTop="15dp"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:background="@drawable/background"
      android:layout_marginBottom="10dp"
      android:id="@+id/search"/>

   <ListView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/list"/>

</LinearLayout>
```

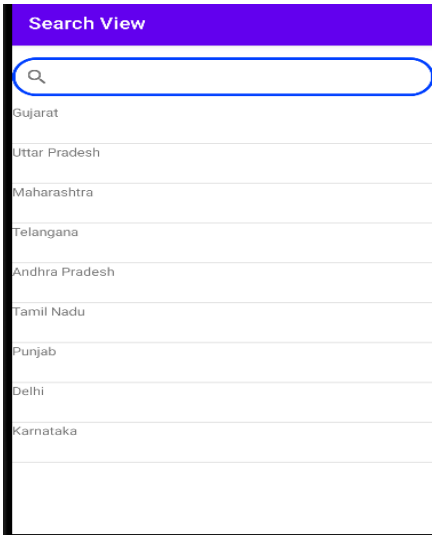
Main_Activity.java

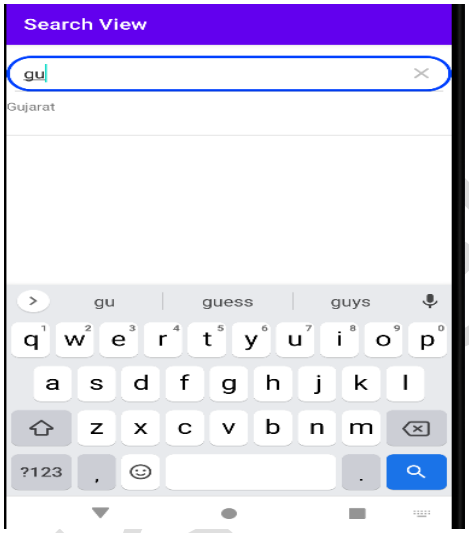
```
package com.example.searchview;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ArrayAdapter;


`
```

```
import android.widget.ListView;
import android.widget.SearchView;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

   SearchView searchView;
   ListView listView;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      searchView = findViewById(R.id.search);
      listView = findViewById(R.id.list);

      final ArrayList<String> arrayList = new ArrayList<>();
      arrayList.add("Gujarat");
      arrayList.add("Uttar Pradesh");
      arrayList.add("Maharashtra");
      arrayList.add("Telangana");
      arrayList.add("Andhra Pradesh");
      arrayList.add("Tamil Nadu");
      arrayList.add("Punjab");
      arrayList.add("Delhi");
      arrayList.add("Karnataka");


      final ArrayAdapter<String> arrayAdapter = new
ArrayAdapter<>(getApplicationContext(),R.layout.support_simple_spinner_dropdown_item,arra
yList);
      listView.setAdapter(arrayAdapter);

      searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
         @Override
         public boolean onQueryTextSubmit(String s) {
            return false;
         }

         @Override
         public boolean onQueryTextChange(String s) {

            arrayAdapter.getFilter().filter(s);
            return false;


`
```

```
        }
    });

  }
}
```

OUTPUT





XML Attribute of Search View

android:iconifiedByDefault	The default state of the SearchView.
android:imeOptions	The IME options to set on the query text field
android:inputType	The input type to set on the query text field.
android:maxWidth	An optional maximum width of the SearchView.
android:queryHint	An optional query hint string to be displayed in the empty query field.

Methods of SearchView

public boolean onQueryTextSubmit(String query)	It searches the query on the submission of content over SearchView editor. It is case dependent.
public boolean onQueryTextChange(String newText):	It searches the query at the time of text change over SearchView editor.

`

2.6 TabLayout and FrameLayout

Tablayout and FrameLayout

- *The TabLayout and FrameLayout are used to create non sliding tabs. By adding the* TabItem *of android support design widget, we can implement the Items of TabLayout.*
- *Example of TabLayout using FrameLayout:*
- In the below example, we are demonstrating the use and behavior of the TabLayout using FrameLayout and Fragment.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="com.example.radioapp.MainActivity">


   <android.support.design.widget.TabLayout
      android:id="@+id/tabLayout"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:background="#7367"
      tools:ignore="MissingConstraints">

      <android.support.design.widget.TabItem
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Home" />

      <android.support.design.widget.TabItem
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="About" />

      <android.support.design.widget.TabItem
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Blog" />
```

`

```
    </android.support.design.widget.TabLayout>

    <FrameLayout
        android:id="@+id/frameLayout"
        android:layout_width="match_parent"
        android:layout_height="455dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tabLayout">

    </FrameLayout>
</android.support.constraint.ConstraintLayout>
```

MainActivity.java

```
package com.example.radioapp;

import android.os.Bundle;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity;
import android.widget.FrameLayout;

public class MainActivity extends AppCompatActivity {
    TabLayout tabLayout;
    FrameLayout frameLayout;
    Fragment fragment = null;
    FragmentManager fragmentManager;
    FragmentTransaction fragmentTransaction;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tabLayout=(TabLayout)findViewById(R.id.tabLayout);
        frameLayout=(FrameLayout)findViewById(R.id.frameLayout);

        fragment = new Home();
        fragmentManager = getSupportFragmentManager();
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.frameLayout, fragment);
        fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);


`
```

```
        fragmentTransaction.commit();

        tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
            @Override
            public void onTabSelected(TabLayout.Tab tab) {
                // Fragment fragment = null;
                switch (tab.getPosition()) {
                    case 0:
                        fragment = new Home();
                        break;
                    case 1:
                        fragment = new About();
                        break;
                    case 2:
                        fragment = new Blog();
                        break;
                }
                FragmentManager fm = getSupportFragmentManager();
                FragmentTransaction ft = fm.beginTransaction();
                ft.replace(R.id.frameLayout, fragment);
                ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
                ft.commit();
            }

            @Override
            public void onTabUnselected(TabLayout.Tab tab) {

            }

            @Override
            public void onTabReselected(TabLayout.Tab tab) {

            }
        });

    }
}
```

Home.java

`

```
package com.example.radioapp;
 import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import app.com.sample.R;

public class Home extends Fragment {
   public Home() {
      // Required empty public constructor
   }
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
      return inflater.inflate(R.layout.fragment_home, container, false);
   }
}
```

fragment_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".Home">
   <!-- TODO: Update blank fragment layout -->
   <TextView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:textAlignment="center"
      android:text="Home Fragment"
      android:textSize="16sp"
      android:textStyle="bold"/>
</FrameLayout>
```

About.java

`

```
package com.example.radioapp;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import app.com.sample.R;

public class About extends Fragment {
   public About() {
      // Required empty public constructor
   }
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
      return inflater.inflate(R.layout.fragment_about, container, false);
   }
}
```

Fragment_about.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".About">
   <TextView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:textAlignment="center"
      android:text="About Fragment"
      android:textSize="16sp"
      android:textStyle="bold"/>
</FrameLayout>
```

Blog.java

`

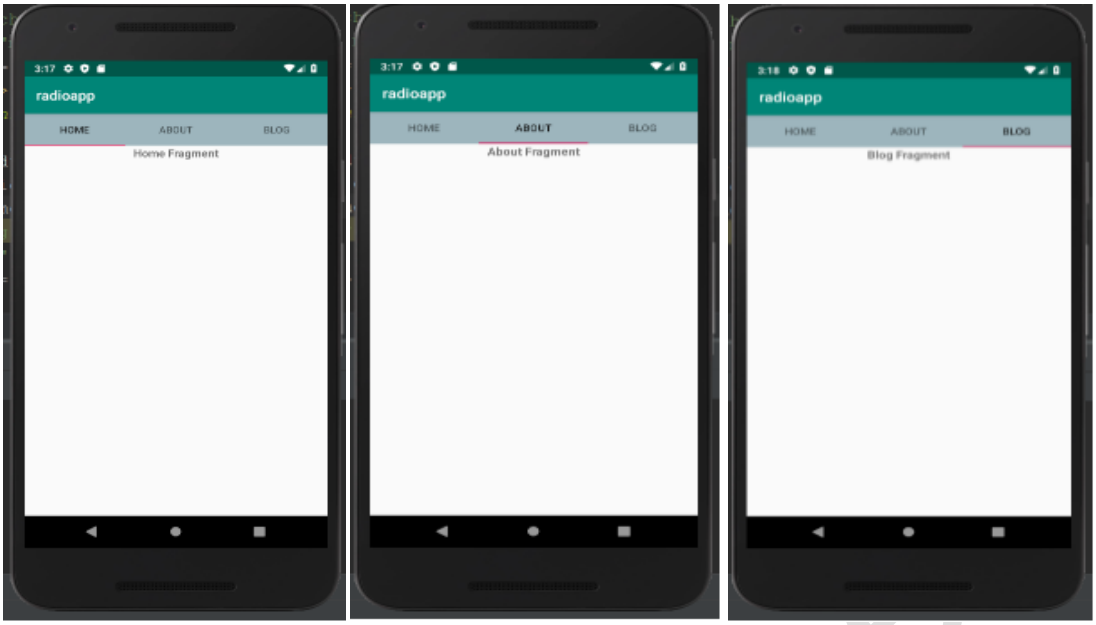
```
package com.example.radioapp;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import app.com.sample.R;

public class Blog extends Fragment {
   public Blog() {
      // Required empty public constructor
   }
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
      return inflater.inflate(R.layout.fragment_blog, container, false);
   }
}
```

Fragement_blog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".Blog">
   <!-- TODO: Update blank fragment layout -->
   <TextView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:textAlignment="center"
      android:text="Blog Fragment"
      android:textSize="16sp"
      android:textStyle="bold"/>
</FrameLayout>
```

OUTPUT

`



`